Clarkson UNIVERSITY

WALLACE H. COULTER SCHOOL OF ENGINEERING
Technology Serving Humanity

MEMORANDUM

From: Bill Jemison
To: Dr. Daniel Tam, ONR
Date: 1/31/2012

Subject: ULI Progress Report 002–Advanced Digital Signal Processing for Hybrid Lidar
FY11 Progress Report (10/1/2011– 12/31/2011)

This document provides a progress report on the project "Advanced Digital Signal Processing for Hybrid Lidar" covering the period of 10/1/2011– 12/31/2011.

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu

Progress Report: ULI - Advanced Digital Signal Processing for Hybrid Lidar

Progress Summary

The technical objective of this project is the development and evaluation of various digital signal processing (DSP) algorithms that will enhance hybrid lidar performance. Practical algorithms must be developed taking into account the underwater propagation channel and the processing requirements for each algorithm as shown in Figure 1.

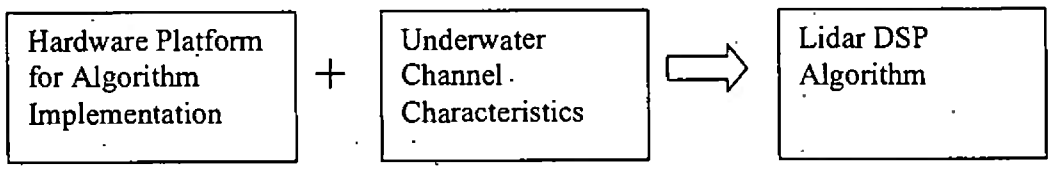


Figure 1. The development of lidar DSP algorithms must take into account hardware implementation and underwater channel characteristics.

The hardware platforms being considered for lidar signal processing are based on software defined radios (SDRs). NAWC has selected a digital hardware platform based on COMBLOCK software defined radio (SDR) modules. Clarkson has investigated several other SDR approaches which include two open source SDR platforms and one commercial platform. We selected the Signal Hound SDR for additional characterization since it is commercially available and has a well-supported Application Programmer's Interface or API. The API allows us to configure the SDR and obtain data in a MATLAB programming environment. The primary activity during this reporting period consisted of evaluating the Signal Hound SDR for lidar signal processing. Specifically we developed software that allows us to configure and collect data from the SDR within the MATLAB programming environment. This will allow us to used MALAB as a DSP development platform. MATLAB is a high-level programming language that is commonly used for signal processing. We then used these programs to test the SDR dynamic range for various SDR gain and sensitivity settings. We have also installed the RANGEINDER underwater channel modeling program and are beginning to understand its capability.

Description of Signal Hound Hardware

The Signal Hound is a software defined radio with the capability of operating as an inexpensive Spectrum Analyzer and Signal Measuring Device. Not only is the Signal Hound able to provide us with the frequency spectrum of a signal but it is also capable of obtaining the signal's I/Q information which we can use for signal processing. The two principal methods of user interaction with the Signal Hound is via its provided Graphical User Interface and via its Application Programmable Interface. The latter method provides the user with a deeper control of the hardware and more flexibility regarding the processing of the data obtained. This section describes the signal path of the SDR.

The following is a simple functional diagram for the Signal Hound:

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu

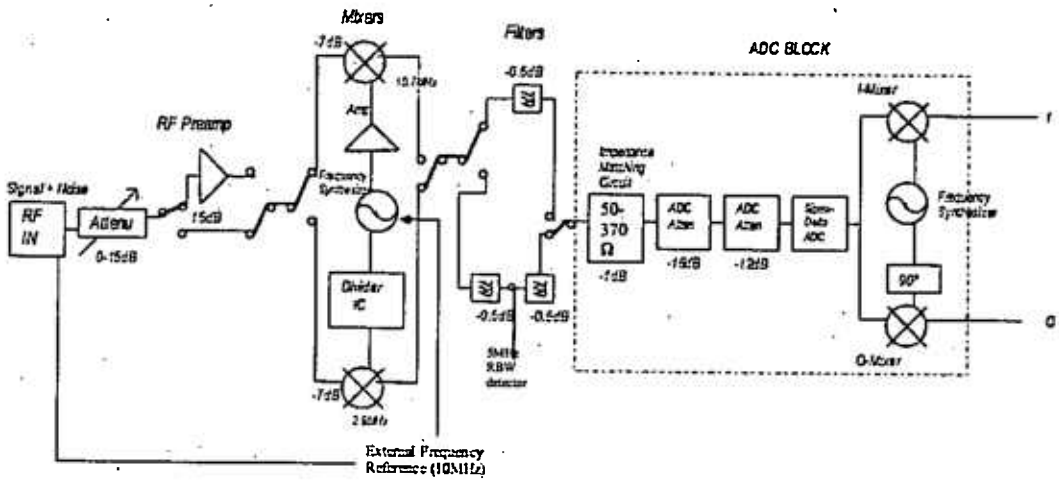


Figure 1. Signal Hound Block Diagram

The system is designed to handle RF signals that are within the range of 0-4.4GHz.

The first component in the RF input chain is a programmable attenuator. It can be varied from 0dB to 15dB in 5dB steps. This is followed by a 15dB gain RF preamplifier that can be set to be switched in or out of the signal path. There are two IF path options which are a 10.7 MHz IF for higher selectivity and a 2.9 MHz IF. Two different user-selectable IF filter paths follow downconversion to IF. The signal then enters the Analog to Digital Conversion (ADC) block. An impedance matching circuit transforms the 50 ohm RF path to the 370 ohm input of the ADC. The signal then goes through two switchable ADC attenuators which have values of -16dB and -12dB. These can be controlled by the user via the API sensitivity input of the Configure command. A sensitivity of 0 gives an attenuation of -28dB, 1 gives an attenuation of -12dB and a sensitivity of 2 gives an attenuation of 0dB. The signal is digitized using a Sigma-Delta ADC with a sample rate of 486.1111 ksps. The output of the ADC is I/Q data which can be decimated from 1 to a value of 16. The user has also control over two ADC clock settings, a 23 1/3 MHz and a 22.5 MHz.

It is noted that mixing occurs in a two-step process. First the RF signal is mixed down to an IF which the user specifies, 10.7MHz or 2.9Mhz. Then at the ADC block performs a second downconversion which may be to DC or to a low IF depending on the center frequency of the Signal Generator and of the Local Oscillator of the Signal Hound. If a low IF is selected a final digital down conversion may be accomplished in software.

It is desired to understand the mapping of RF input power to digital data. The Signal Hound's ADC is the AD9864 from Analog Devices. It is a 16- bit ADC with a maximum power input of -19dBm and a minimum power of -114dBm (according to its specsheets which show a 95dB of dynamic range).

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu

Converting these two power values into voltages we obtain peak voltages of V_H = 96.5mV and of V_L ~0mV. From these we can obtain the ADC quantization level $q = \frac{V_H - V_L}{2^{16} - 1} \cong 1.4\,\mu V$.

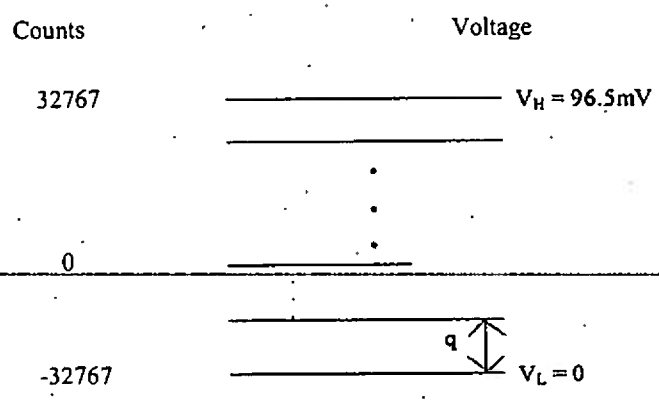


Figure 2. ADC Quantization level

The Signal Hound ADC's Raw output is the IQ data in counts, from -32767 to +32767 and these can be related to voltage. To convert from counts to a voltage one must first add 32767 to the number of counts (to account for the negative counts) and then multiply this result by the quantization level q. Dividing this voltage by the square root of 2 gives us an rms voltage which we can square and divide by the ADC Resistance (R_{ADC} = 370 ohms) to obtain the power of the signal in watts. To convert to units of dBm, take the log of the power in watts, multiply by 10 and add 30 to the answer.

Signal Hound Spectrum Analyzer GUI

The Signal Hound Spectrum Analyzer GUI Test Equipment Plus is one method of communication between the user and the Signal Hound platform over a USB connection. The user interacts with the Signal Hound via commands sent using the GUI's Control Panel and Menu (See the figure below). There are several gain options available for the user to adjust, some of them include allowing the user to add attenuation (0 -15dB) and toggle the preamplifier on or off. The GUI also has a number of typical spectrum analyzer settings that allow the user to set the center frequency and the span, adjust the amplitude reference level of the display, place a marker, search for signal's peak, set the resolution bandwidth, set reference frequency, set trigger controls (continuous, single, free run, video), among several other features. The GUI will automatically adjust the gain settings based on these inputs to provide the appropriate dynamic range.

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu

Signal Hound Application Programming Interface (API)

The Signal Hound's API provides the user with the freedom to design their own specialized applications to use with the Signal Hound. Just like the GUI, the user is able to send commands to the Signal Hound as well as receive information from it. Via the API the user can have the same capabilities as with the GUI but unlike the GUI, using the API gives a deeper level of control to the user and it allows the user to collect, process and store the data in any format desired. It also gives more freedom as to what processing one wishes to do with the data collected. It consists of three main files, a dynamic link library -the SH_API.dll, a header file- SH_API.h, and a library file- SH_API.Lib. The API is available for windows, Linux, along with a Matlab and LabView version.

The first two API commands that must be run regardless of the user application are the initialization command and the configuration command. The initialization command calibrates the Signal Hound device and it takes approximately twenty seconds to run. The configure command serves to set up the Signal Hound to receive the RF signal. Using the configure command, the user is able to set the preamplifier, the attenuation, the mixer band, sensitivity, decimation, intermediate frequency path, and the ADC clock.

There are several different API functions that allow us to do different tasks with our RF signal. For example the API "Slow Sweep" and "Fast Sweep" functions allows us to take the FFT of the RF signals while the "Get I/Q Data Packet" gives us the RF signal's I/Q data.

Streaming of I/Q data using the Signal Hound's API

To stream I/Q data using the Signal Hound's API the following steps are used:

1) Initialize the Signal Hound
2) Configure the Signal Hound
3) Setup LO
4) Start Streaming I/Q data
5) Recurrently Get Streaming Packet
6) Stop Streaming I/Q data

The first step that must take place is the initialization of the Signal Hound using the API function SHAPI_Initialize(). This function takes about 20 seconds to execute. If one were using Multiple Signal Hounds the function that would be needed is the SHAPI_InitializeNext() and SHAPI_SelectDevice(deviceToSelect). The SHAPI_InitializeNext() initialized the next Signal Hound and the SHAPI_SelectDevice(deviceToSelect) function allows the user to switch between devices. The Argument 'deviceToSelect' specifies from 0 to 7 (0 being the first device and 1 being the next and so on) the Signal Hound currently selected.

The second step that must take place following the initialization step and prior to any steps is the configuration of the Signal Hound. This uses the API function SHAPI_Configure(attenVal, mixerBand, sensitivity, decimation, IF_Path, ADC_clock). This is a powerful command that allows us to have

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu

control over the Signal Hound's gains and attenuation settings. The following provides brief information about each of these arguments.

attenVal – Attenuator setting. Can be adjusted from 0 to 15 dB in 5dB steps. 10dB is the default.

mixerBand – Set to 0 if your RF input frequency is below 150 MHz otherwise set to 1.

Sensitivity – Controls ADC input attenuation settings. Set to 0 for -28dB attenuation, set to 1 for a -12dB attenuation or set to 2 for 0 dB attenuation.

Decimation – A number between 1 and 16 by which the sampling frequency is divided. the sampling frequency is 486.1111Ksps.

IF Path – Set to 0 for 10.7MHz Intermediate frequency or set to 1 for 2.9MHz IF. The default is 0 (IF = 10.7 MHz).

ADC clock – Can be set to 0 for 23.33 MHz or 1 for 22.5 MHz (if signal is a multiple of 23.33MHz). The default is 0 (ADC clock = 23.33MHz).

The third step is to set up the Local Oscillator (LO) using the SHAPI_SetupLO(centerFreq, mixMode, deviceNum) function.

centerFreq – This argument is passed by reference. The LO is chosen to be as near as possible to the user specified center frequency. The function sets the centerFreq to the actual center frequency.

mixMode – Set to 0 for low side injection, set to 1 for high side injection (this is the default).

deviceNum – currently selected Signal Hound. A deviceNum of -1 indicates the currently selected Signal Hound device.

The fourth step is to start the data streaming process by using the SHAPI_StartStreamingData(deviceNum).

Next, the command SHAPI_GetStreamingPacket(*bufI, *bufQ, deviceNum) gets a packet of 4096 I and Q samples (bufI, bufQ) every time this function is called. These values are in counts.

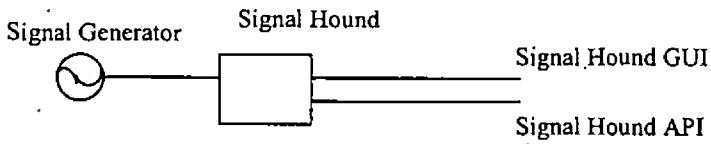
Lastly, the command SHAPI_StopStreamingPacket(deviceNum) is executed to end the data streaming process.

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu

Dynamic Range Test

The instantaneous dynamic range of the Signal Hound was measured by plotting the power output measured by the Signal Hound versus the input power that was put into it. The measurements continued until a 1dB compression point was achieved.

Set up:



Signal Hound Noise Floor:

Thermal Noise Floor

$P = KTB$

P, power in watts
K, Boltzmann's Constant (1.38×10^{-23} J/K)
B, Bandwidth in Hertz

At room temperature (290K), this results in $P_{dBm} = -174$ dBm/Hz.

$P_{dBm} = -174 + 10*\log_{10}(B)$

B, is the bandwidth in Hz.

Quantization Noise Floor

Quantization noise level in watts $= q^2 / 12$

q, quantization level

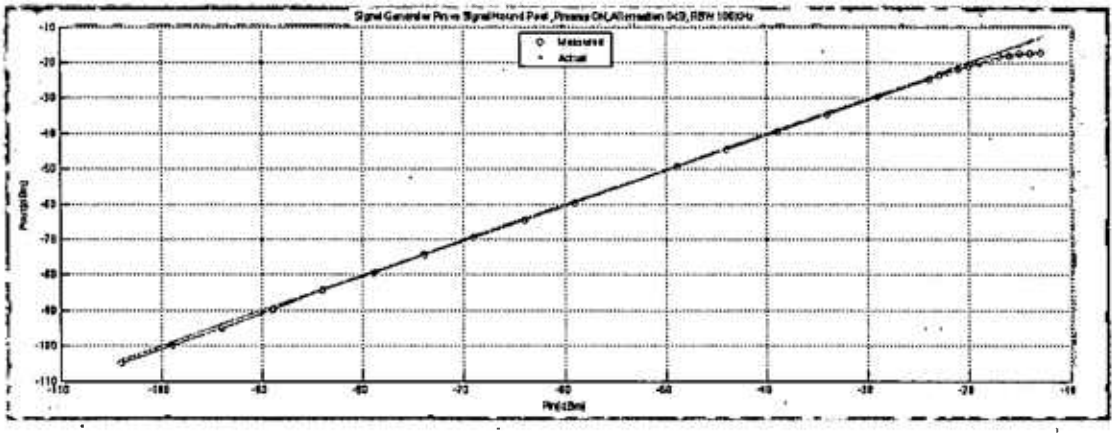
Quantization noise in dBm $= 10*\log_{10}(q^2/12) + 30$

Using our ADC's quantization level q = 1.47μV, the calculated Quantization Error in dBm is equal to about -97 dBm.

Dynamic Range Test using the spectrum analyzer GUI

The Signal Hound automatically adjust the gain to avoid compression for the currently given amplitude reference level. When dynamic range test measurements were being made we adjusted the amplitude reference level in a way that would not change while we had assumed a 96 dB dynamic range. Therefore since the reference level did not change and we did not exceed the amplitude reference level, the automatic gain was not employed. The power from the Signal Generator (Pin) was varied and the power measured using the Signal Hound's GUI (Pout) was recorded. We can observe a little over 80dB of instantaneous dynamic range.

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu



The 1dB compression point occurs at Pin = -19dBm.

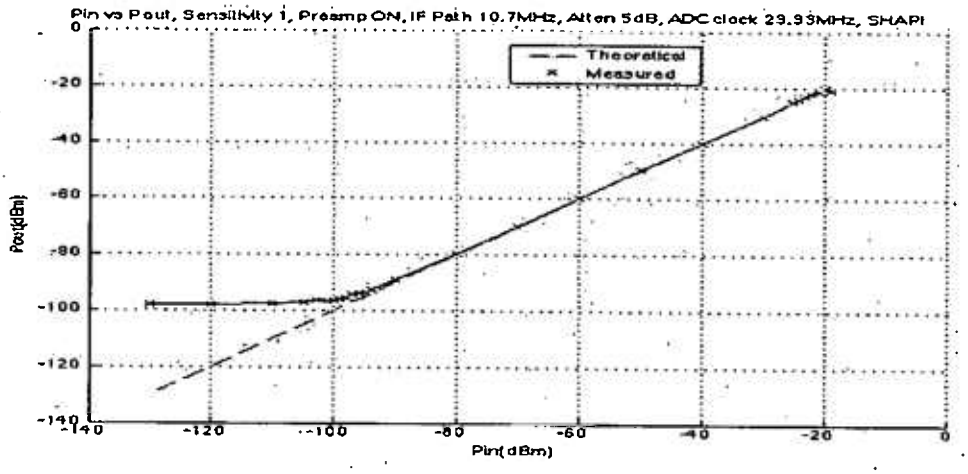
Dynamic Range Measruement Using the API

The same measurement was carried using the Signal Hound API following the procedure described earlier for obtaining I/Q data and converting this data to a power in dBm.

-The Signal Generator's power (Pin) was varied.
-At each power setting, the I/Q data was obtained (about 4,861 I/Q raw count data pairs collected) by the procedure indicated in section ID. The settings used are indicated below. To implement this procedure, a Matlab code that called the API C functions was used.
-The mean of the I data and the mean of the Q data were computed.
- The amplitude was calculated by taking the square root of the addition of I squared and Q squared.
- This value in counts was converted to power in dBm by the procedure explained earlier in this memo (section IB).
- This value in dBm would then be our Pout measured by the Signal Hound.

The settings used: Sensitivity 1, Preamp ON(+15dB), IF Path 10.7MHz, Attenuation 5dB, ADC clock 23.33MHz. We again see about 80dB of instantaneous dynamic range therefore agreeing with the Signal Hound GUI.

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu



We see the 1dB compression point occurs at around Pin = -19dBm.

Summary

In summary, we have successfully configured the SDR using the API in a MATLAB environment and we are now able to capture real-time data. Experiments were performed to measure the instantaneous dynamic range of the SDR using the API and the vendor supplied GUI. This work lays the foundation which will allow us to implement custom signal processing algorithms using the SDR. Also, we have begun to use the RANGEFINDER simulator to help understand and model the underwater lidar channel.

William D. Jemison, Professor and Chair, PO Box 5720, Clarkson University, Potsdam, NY 13699-5720
315-268-6509, Fax 315-268-7600, wjemison@clarkson.edu